正课:

1. \*\*\*\*Function

\*\*\*\*作用域链

\*\*\*\*\*闭包

2. \*\*\*面向对象OOP

封装，继承，多态

1. \*\*\*\*Function

作用域(scope)和作用域链:

什么是: 用途: 一个变量的可用范围

本质: 其实是一个保存变量的对象

为什么: 避免不同范围的变量间互相干扰

包括: 2级:

1. 全局作用域: window

保存全局变量: 优: 可重复使用，随处可用

缺: 全局污染

2. 函数作用域: 临时创建的活动对象AO

保存局部变量: 优: 不污染全局

缺: 不可重用，仅函数内可用

强调: 局部变量包含2中:

1. 在函数内var的

2. 参数变量也是局部变量

原理: 四个阶段:

1. 程序开始执行前:

在内存中创建执行环境栈(数组):

执行环境栈: 依次保存正在调用的函数的数组

函数调用后，会出栈

在栈中首先添加浏览器主函数的调用

创建全局作用域对象window

主函数引用全局作用域对象window

2. 定义函数时:

在window中用函数名创建变量

在window外创建函数对象保存函数的内容

函数名变量通过地址引用函数对象

函数对象用隐藏的scope属性，引用回自己诞生的作用域对象。——万一运行时缺东西，可以找诞生的作用域要！

3. 函数调用时:

在执行环境栈中添加当前函数调用

为本次函数调用创建活动对象AO:

活动对象: 专门保存本次函数调用所需的局部变量的对象——就是函数作用域对象

在活动对象中保存函数的局部变量

活动对象通过隐藏的parent属性引用window

变量的使用顺序: 先用AO中的局部变量，局部没有，才去window找。

4. 函数调用后:

函数的调用从ECS中出栈

导致: 函数作用域对象AO释放

导致: 局部变量一同释放！

作用域链: 由多级作用域对象，逐级引用形成的链式结构

作用: 存储所有变量(局部变量和全局变量)

控制着变量的使用顺序，先局部再全局

\*\*\*\*\*闭包:

什么是: 即重用一个变量，又保护变量不被篡改的一种机制

为什么:

全局变量: 优: 可重用

缺: 造成全局污染，易被篡改

局部变量:

优: 仅函数内可用，不会污染全局，也不会被篡改

缺: 不可重用

何时: 只要希望重用一个变量，又保护变量不被污染时

如何: 3步: 鉴别是否形成闭包的特征

1. 用外层函数包裹，要保护的变量和使用变量的内层函数

2. 外层函数返回内层函数

3. 调用者，调用外层函数，获得内层函数对象

缺: 闭包比普通函数占用更多内存！

解决: 用完后立刻释放！

getNum=null 导致: 内层小函数对象释放

导致: outer的AO释放